WHAT IS CLAIMED IS:

1. A method of annotating a computer program, comprising:

a) applying a program checking tool to the computer program to produce one or more warnings;

b) mapping at least one of said warnings into at least one annotation modification;

c) modifying the computer program in accordance with said at least one annotation modification so that the number of annotations in the computer program changes, thereby producing a modified computer program;

d) repeating each of steps a), b) and c) until no warnings produced in step a) are suitable for mapping into an annotation modification; and

e) providing a user with the modified computer program in which is found at least one annotation.

2. The method of claim 1 wherein at least a subset of said warnings are warnings about potential misapplications of primitive operations in the computer program.

3. The method of claim 2 wherein, prior to said mapping, said warnings about potential misapplications of primitive operations in the computer program are identified, and said modifying comprises inserting into the computer program at least one annotation that is produced by mapping at least one of said warnings about potential misapplications of primitive operations into an annotation modification.

4. The method of claim 1 wherein, prior to said applying, a candidate set of heuristically derived annotations is inserted into the computer program.

5. The method of claim 4 wherein at least a subset of said warnings are warnings about inconsistencies between the computer program and one or more of the annotations.

6. The method of claim 5 wherein said warnings about inconsistencies between the computer program and one or more of the annotations are identified, and said modifying

comprises removing from the computer program one of said heuristically derived annotations identified by said at least one annotation modification.

7. The method of claim 4 wherein said set of candidate annotations comprises a candidate invariant for a variable f.

8. The method of claim 4 wherein said set of candidate annotations comprises at least one candidate precondition for a procedure in said computer program.

9. The method of claim 4 wherein said set of candidate annotations comprises at least one candidate postcondition for a procedure in said computer program.

10. The method of claim 7 wherein said candidate invariant is of the form f != null.

11. The method of claim 7 wherein said candidate invariant comprises an expression that includes a comparison operator.

12. The method of claim 11 wherein said comparison operator is selected from the group consisting of: <, <=, =, !=, >= and >.

13. The method of claim 11 wherein said expression includes an operand selected from the group consisting of: a variable declared earlier in a same class of the computer program; any one of the constants -1, 0, 1; and a constant dimension in an array allocation expression in the computer program.

14. The method of claim 1 wherein at least one of said warnings includes an explanation.

15. The method of claim 1 wherein at least one of said annotations in said modified computer program includes an explanation.

16. The method of claim 6 wherein said removing comprises commenting out one of said heuristically derived annotations from the computer program.

17. The method of claim 16 wherein said removing additionally comprises adding an explanatory comment into one of said heuristically derived annotations from the computer program.

18. The method of Claim 3 wherein said annotation includes an explanatory comment.

19. The method of Claim 1 wherein said program checking tool is a type checker.

20. The method of Claim 1 wherein said program checking tool is an extended static checker.

21. The method of Claim 1 wherein said program checking tool comprises a verification condition generator and a theorem prover.

22. A computer program product for use in conjunction with a computer system, the computer program product comprising a computer readable storage medium and a computer program mechanism embedded therein, the computer program mechanism comprising:

a program checking tool for analyzing a computer program to produce one or more warnings;

at least one warning mapper for mapping at least one of said warnings into at least one annotation modification;

a program updater for updating the computer program in accordance with the annotation modification so that the number of annotations in the computer program changes; and

control instructions for repeatedly invoking the program checking tool, warning mapper and program updater until no warnings produced by the program checking tool are suitable for mapping into an annotation modification.

23. The computer program product of claim 22 wherein at least a subset of said warnings are warnings about potential misapplications of primitive operations.

24. The computer program product of claim 23, wherein said at least one warning mapper includes instructions for identifying said warnings about potential misapplications of primitive operations, and the program updater includes instructions for inserting into the computer program an annotation that the warning mapper produces by mapping at least one of said warnings about potential misapplications of primitive operations into an annotation modification.

25. The computer program product of claim 22, further including a heuristic annotation generator for generating and inserting a candidate set of heuristically derived annotations into the computer program.

26. The computer program product of claim 25 wherein at least a subset of said warnings are warnings about inconsistencies between the computer program and one or more of the annotations.

27. The computer program product of claim 25 wherein the warning mapper includes instructions for identifying said warnings about inconsistencies between the computer program and one or more of the annotations, and the program updater includes instructions for removing from the computer program one of said heuristically derived annotations identified by said annotation modification.

28. The computer program product of claim 25 wherein said candidate set of annotations comprises a candidate invariant for a variable f.

29. The computer program product of claim 28 wherein said candidate invariant comprises an expression that includes a comparison operator.

30. The computer program product of claim 29 wherein said comparison operator is selected from the group consisting of: <, <=, =, !=, >= and >.

31. The computer program product of claim 29 wherein said expression includes an operand selected from the group consisting of: an earlier declared variable in a same class of

the computer program; any one of the constants -1, 0, 1; and a constant dimension in an array allocation expression in the computer program.

32. The computer program product of claim 27 wherein said instructions for removing comprise instructions for commenting out one of said heuristically derived annotations from the computer program.

33. The computer program product of claim 22 wherein said program checking tool is a type checker.

34. The computer program product of claim 22 wherein said program checking tool is an extended static checker.

35. The computer program product of claim 22 wherein said program checking tool comprises a verification condition generator and a theorem prover.

36. A system for annotating a computer program with at least one annotation, the system comprising:

at least one memory, at least one processor and at least one user interface, all of which are connected to one another by at least one bus;

wherein said at least one processor is configured to:

annotate the computer program with at least one annotation;

apply a program checking tool to the computer program to produce one or more warnings;

map at least one of said warnings into at least one annotation modification;

modify the computer program in accordance with the annotation modification so that the number of annotations in the computer program changes; and

repeat applying the program checking tool, mapping said warnings and modifying the program until no warnings produced by the program checking tool are suitable for mapping into an annotation modification.

37. The system of claim 36 wherein at least a subset of said warnings are warnings about potential misapplications of primitive operations.

38. The system of claim 37, wherein said at least one processor identifies said warnings about potential misapplications of primitive operations, and inserts into the computer program an annotation that a warning mapper produces by mapping at least one of said warnings about potential misapplications of primitive operations into an annotation modification.

39. The system of claim 36, wherein said at least one processor further causes a heuristic annotation generator to generate a candidate set of heuristically derived annotations.

40. The system of claim 38 wherein at least a subset of said warnings are warnings about inconsistencies between the computer program and one or more of the annotations.

41. The system of claim 39 wherein the at least one processor removes from the computer program one of said heuristically derived annotations identified by said annotation modification.

42. The system of claim 39 wherein said candidate set of annotations comprises a candidate invariant for a variable f.

43. The system of claim 42 wherein said candidate invariant comprises an expression that includes a comparison operator.

44. The system of claim 43 wherein said comparison operator is selected from the group consisting of: <, <=, =, !=, >= and >.

45. The system of claim 43 wherein said expression includes an operand selected from the group consisting of: an earlier declared variable in a same class of the computer program; any one of the constants -1, 0, 1; and a constant dimension in an array allocation expression in the computer program.

46. The system of claim 41 wherein the at least one processor comments out one of said heuristically derived annotations from the computer program.

47. The system of claim 36 wherein said program checking tool is a type checker.

48. The system of claim 36 wherein said program checking tool is an extended static checker.

49. The system of claim 36 wherein said program checking tool comprises a verification condition generator and a theorem prover.

50. A method of annotating a computer program, comprising:

a) applying a program checking tool to the computer program to produce one or more warnings about potential misapplications of primitive operations in the computer program;

b) mapping at least one of said warnings into at least one annotation modification;

c) inserting into the computer program said at least one annotation modification, thereby producing a modified computer program;

d) repeating each of a), b) and c) until no new warnings are produced in a) that are suitable for mapping into an annotation modification; and

e) providing a user with the modified computer program in which is found at least one annotation.

51. A method of annotating a computer program, comprising:

a) inserting a candidate set of annotations into the computer program by employing a heuristic analysis of the computer program;

b) applying a program checking tool to the computer program to produce one or more warnings about inconsistencies between the computer program and one or more of the annotations;

c) mapping at least one of said warnings into at least one annotation modifications;

d) removing from the computer program an annotation, from said set of candidate annotations, that is mentioned by at least one of said warnings, thereby producing a modified computer program;

e) repeating each of b), c) and d) until no new warnings are produced in b) that are suitable for mapping into an annotation modification; and

f) providing a user with the modified computer program in which is found at least one annotation.